Finding Lane Lines on the Road

The goals / steps of this project are the following:

- Make a pipeline that finds lane lines on the road
- Reflect on your work in a written report

Reflection

Pipeline Description

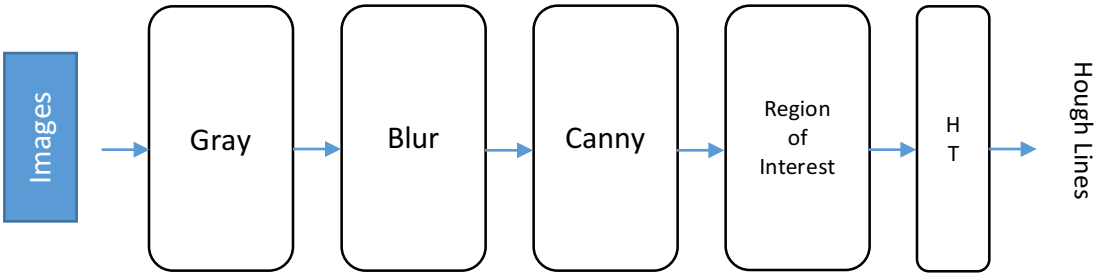

Fig1. Pipeline Implemented

Using the helper functions in the notebook, images are converted to gray scale, blurred and fed into the Canny function to detect edges, once the edges have been detected, these images are fed into the Hough Function. The Probabilistic Hough transform provides a set of coordinate points (min and max) the ones that specify the start and end of a detected line in the images. Lastly, the function to draw and weight image are invoked and these ones output the detected lines in the actual image where the lanes are.

Example images:

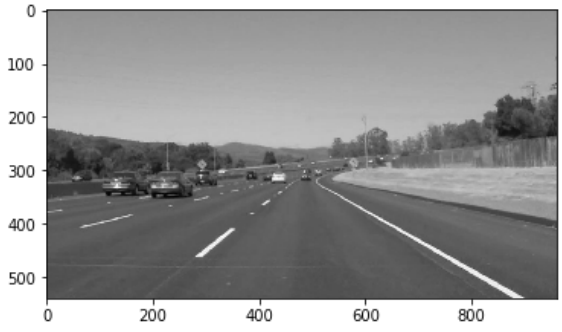

Fig2. Gray

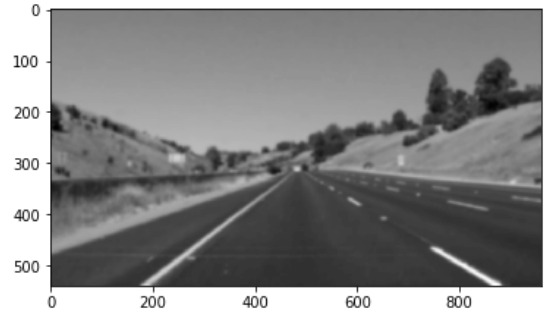

Fig2. Blur

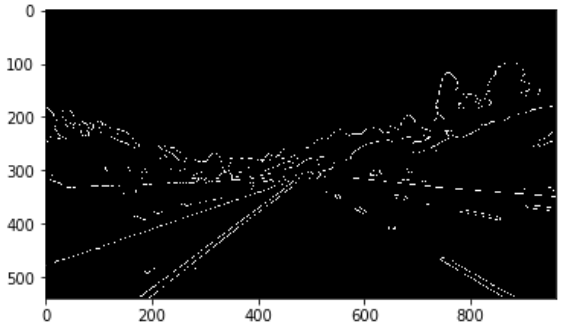

fig3. Canny output

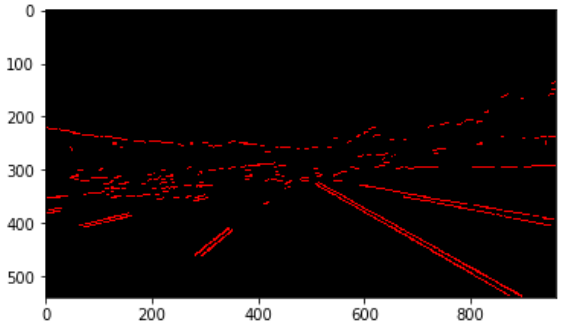

fig4. HT lines

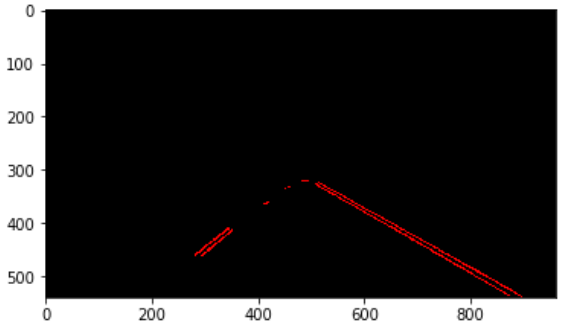

fig 5. Region of interest

Improvements

With the provided *Draw_lines* function, it is possible to observe some results in the detected lanes, nevertheless, the function does not extrapolate and neither draw a complete single line, therefore some improvements were made:

1. The lines obtained from probabilistic Hough Transform were used to detect a slope and interception.
2. Based on the slope sign (negative or positive) the lines were classified as to belong to the left or right side of the image road. It was taken into account that the coordinates plane where the image resides is upside down (Y axis increasing towards the bottom of the image and the coordinate (0,0) in the top left)
3. With the slope (*m*), intercept (*b*) and the set of coordinates of the region of interest of the image (y1, y2), I proceed to find the x1, x2 points that belong to the line described by the slope and intercept found, such points are passed as arguments to the **OpenCV line and a complete single line then is obtained.**
4. Although a complete single line was obtained as described in the point 3, this one was not stable and presented a lot of jittering, probably because of the presence of outliers points, to address this issue, in a separate array the average slope was calculated for each line.
5. With the average slope for each left or right line, a simple conditional was applied in order to keep in track the drawn line and avoid the usage of points that were too distant from the obtained Probabilistic Hough Transform lines, by applying this conditional, it is possible to observe in the videos two things:
   - The jittering was reduced, drawing a more stable line.
   - The single line obtained doesn't deviate that much from the detected lane lines and it is drawn within an acceptable range that is not too far from the original Hough Lines and not too far from the average of all the points that constitute all the lines.

Example output

Fig6. The detected lane line is drawn with less jitter and around the actual lane

Potential Shortcomings and suggestions

1. Curves, the line is too rigid, in the presence of curvatures the jitter and drawn line will deviate a lot.
2. Different line conditions that would require more sophisticated image filter

References

http://ottonello.gitlab.io/selfdriving/nanodegree/python/line%20detection/2016/12/18/extrapolating_lines.html